Examiner-Initiated Interview Summary	Application No.	Applicant(s)	
	10/706,776	RICHARDSON, JOHN J.	
	Examiner	Art Unit	
	Abdou Karim Seye	2194	

All Participants:

(1) *Marisa J. Zink*.

(2) *Abdou Karim Seye*.

Status of Application: _____

(3) _____.

(4) _____.

Date of Interview: _____

Time: _____

Type of Interview:
- ☒ Telephonic
- ☐ Video Conference
- ☐ Personal (Copy given to: ☐ Applicant ☐ Applicant's representative)

Exhibit Shown or Demonstrated: ☐ Yes ☐ No
If Yes, provide a brief description:

Part I.

Rejection(s) discussed:
Possible 112 rejection for claim 6

Claims discussed:

Prior art documents discussed:

Part II.

SUBSTANCE OF INTERVIEW DESCRIBING THE GENERAL NATURE OF WHAT WAS DISCUSSED:
Examiner discussed with applicant to amend independent claims 22 and 26 to include limitations covered in independent claim 1.

Part III.

☒ It is not necessary for applicant to provide a separate record of the substance of the interview, since the interview directly resulted in the allowance of the application. The examiner will provide a written summary of the substance of the interview in the Notice of Allowability.

☐ It is not necessary for applicant to provide a separate record of the substance of the interview, since the interview did not result in resolution of all issues. A brief summary by the examiner appears in Part II above.

_____ (Examiner/SPE Signature)

_____ (Applicant/Applicant's Representative Signature – if appropriate)

PROPOSED AMENDMENTS TO THE CLAIMS

This listing of claims will replace all prior versions of claims in the application:

Listing of Claims:

1. (Currently Amended) A computer-based event handling system, recorded on a computer-readable medium and capable of execution by a computer, comprising:

a computer process that executes the following software components;

a framework component that supplies classes of objects that can raise events;

a synchronization component that controls in part synchronization of access to data based on categorization of at least one of objects and instances defined by the framework, and provides automatic serialization for events raised by the classes of objects, the events are managed by the framework or synchronization component to allow one or more aspects of the events to occur in a one-time manner and in accordance with a serialized process; and

a configuration component that enables a ~~the~~ client component to disable or enable automated serialization and synchronization;

wherein the framework is supplied by an operating system or as a library for use by the operating system, wherein at least one of the framework and synchronization component automatically manages or serializes the events in order that a client component can process client-specific tasks;

wherein the framework component allows a device driver to allocate additional memory, such that the device driver stores its working data in this allocated object extension memory and relies on the framework component for concurrency management and serialization.

2. (Canceled)

3. (Previously Presented) The system of claim 1, the client component is a device driver.

4. (Canceled)

5. (Previously Presented) The system of claim 1 the events are processed as part of a request.

6. (Previously Presented) The system of claim 1 the object provides a handle to enable the client component to manipulate the object and request additional local memory to be allocated for processing client tasks.

7. (Canceled)

8. (Previously Presented) The system of claim 6, the configuration component includes one or more Application Programming Interfaces to facilitate selection of serialization and synchronization features.

9. (Original) The system of claim 1, the framework component provides events to a device driver though a series of callback functions registered by the device driver.

10. (Original) The system of claim 9, the device driver generally queues a handler or starts Input/Output (I/O) on the handler, marking state, and then returns.

11. (Original) The system of claim 1, the framework component is structured in accordance with state full objects that allow a device driver to register events, and provide API's.

12. (Original) The system of claim 1, the events are associated with a pipeline.

13. (Original) The system of claim 12, the pipeline includes at least one of a dispatch component, a plug and play component, a device model component, and an Input/Output component.

14. (Original) The system of claim 12, the pipeline employs a series of stages that a request traverses in the processing of a Driver Model request or an I/O Request Packet (IRP).

15. (Original) The system of claim 14, further comprising objects within each stage of the pipeline that raise an event to a device driver through an event callback to allow the driver to have control of which action to take, or provide some default action that may result in the requests completion, or forwarding to the next stage.

16. (Original) The system of claim 15, further comprising a processing component to determine whether a request reaches an end of the pipeline without being accepted by the device driver or by automatic processing in a respective stage, otherwise a default processing of the request occurs.

17. (Original) The system of claim 16, the default processing depends on whether the driver is configured as a Filter Driver in which the request is forwarded to a next lower device driver, or a non-filter driver in which the request is completed with a STATUS_INVALID_DEVICE_REQUEST.

18. (Original) The system of claim 1, further comprising one or more data packets that are employed to facilitate communications between the components *via* at least one of a local processing system, a local network system, and a remote network system.

19. (Original) A computer readable medium having computer readable instructions stored thereon for implementing the framework component and the synchronization component of claim 1.

20. (Canceled)

21. (Canceled)

22. (Currently Amended) A method to facilitate event processing in accordance with an operating system, comprising:

defining at least one rule for processing an event;

defining at least one object responsive to the event, the object categorized according to a framework, wherein the framework is supplied by an operating system or as a library for use by the operating system;

automatically serializing the event in accordance with at least one other event;

configuring automatic serialization of the event;

managing the event by the framework or a synchronization component to allow one or more aspects of the event to occur in a one-time manner and in accordance with a serialized process;

disabling or enabling automated serialization and synchronization; ~~and~~

~~allocating additional memory and storing working data in this allocated object extension memory~~

automatically managing or serializing the event, wherein at least one of the framework and synchronization component automatically manages or serializes the events in order that a client component can process client-specific tasks; and

allowing a device driver to allocate additional memory, such that the device driver stores its working data in this allocated object extension memory and relies on the framework component for concurrency management and serialization.

23. (Original) The method of claim 22, further comprising automatically serializing the event in accordance with at least one of an operating system process and the framework.

24. (Original) The method of claim 22, further comprising enabling a client component to disable automatic serialization of events.

25. (Original) The method of claim 24, further comprising providing a local memory to facilitate client-specific processing.

26. (Currently Amended) A computer readable medium having a data structure stored thereon, comprising:

a first data field related to a framework object having a handle associated therewith to facilitate data access;

a second data field that relates to a client component that communicates *via* the handle and receives events from the framework object; and

a third data field that automatically serializes the events, and enables the client component to disable or enable automated serialization and synchronization, the events are managed by the framework object or a synchronization component to allow one or more aspects of the events to occur in a one-time manner and in accordance with a serialized process;

wherein the framework object is supplied by an operating system or as a library for use by the operating system, wherein at least one of the framework object and synchronization component automatically manages or serializes the events in order that a client component can process client-specific tasks;

wherein the first data field allows a device driver to allocate additional memory, such that the device driver stores its working data in this allocated object extension memory and relies on the framework object for concurrency management and serialization.

27. (Original) The computer readable medium of claim 26, further comprising at least one Application Programming Interface (API).

28. (Original) The computer readable medium of claim 27, the API includes at least one of a scope object, a scope device and a scope specified data structure.

29 (Original) The computer readable medium of claim 27, the API includes at least one of an execution level passive, an execution level dispatch, and an execution level specified data structure.

30. (Original) The computer readable medium of claim 27, further comprising an acquire lock data structure and a release lock data structure.